NORTHWESTERN POLYTECHNIC UNIVERSITY

Linear regression using Normal Equation

Prepared For:

Mr. Henry Chang
Machine Learning and Business Intelligence CS550
Summer 2021
Northwestern Polytechnic University

Prepared By:

Ms. Nagalla, Santhi Sree ID:19568

Table of Contents

The Normal Equation

To find the value of $\boldsymbol{\theta}$ that minimizes the cost function, there is a *closed-form solution*—in other words, a mathematical equation that gives the result directly. This is called the *Normal Equation*.

$$\hat{\theta} = (X^\intercal X)^{-1} X^\intercal y$$

In above equation:

- $\hat{\theta}$ is the value of $\boldsymbol{\theta}$ that minimizes the cost function.
- $\mathbf{y}$ is the vector of target values containing $y^{(1)}$ to $y^{(m)}$.

Read Data from CSV

- Let's read some linear-looking data to test this equation from CSV File.

```
import numpy as np

import pandas as pd

from google.colab import files

uploaded  = files.upload()
```

```
import io

abalone = pd.read_csv(
io.BytesIO(uploaded['abalone_train.csv']),

    names=["Length",
"Diameter", "Height",
"Whole weight", "Shucked
weight","Viscera weight",
"Shell weight", "Age"])
```

Assign the data to a variable

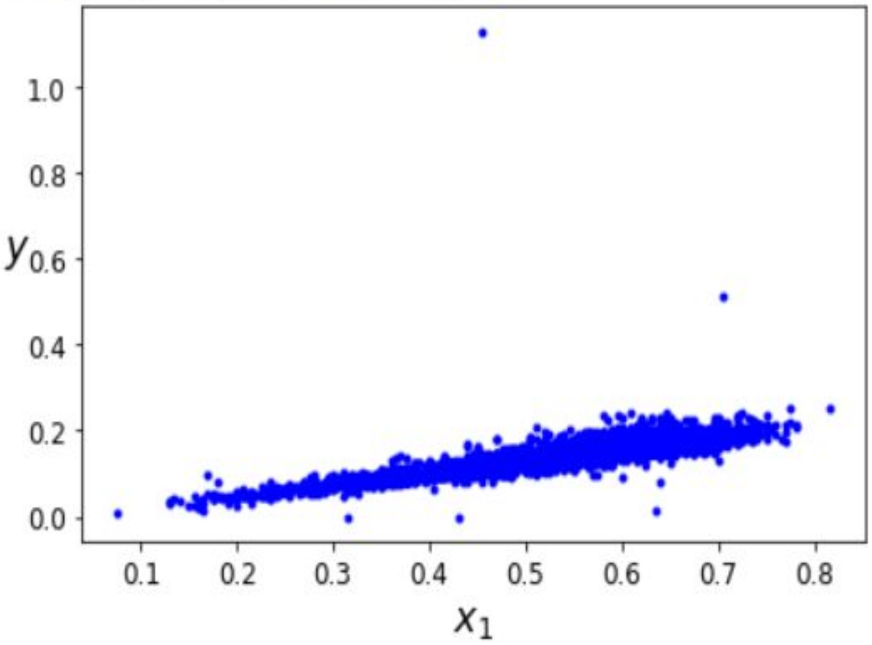
```
X1 = abalone["Length"]
X2 = np.array(X1)
//output:(is a 1 dimensional column array)
X = X2.reshape(-1, 1)
y1 = abalone["Height"]
y2 = np.array(y1)
//output:(is a 1 dimensional column array)
y = y2.reshape(-1, 1)
X,y
```

```
(array([[0.435],[0.585],[0.655],
    ...,
    [0.53 ],[0.395],
    [0.45 ]]), array([[0.11 ],
    [0.125],[0.16 ],
    ...,
    [0.13 ],[0.105],[0.12 ]]))
```

Plot and Save Data

```
//Plot and save the data
plt.plot(X, y, "b.")
plt.xlabel("$x_1$", fontsize=18)
plt.ylabel("$y$", rotation=0,
fontsize=18)
save_fig("generated_data_plot")
plt.show()
```

Saving figure generated_data_plot



Compute ˆθ

- let's compute ˆθ using the Normal Equation. We will use the inv() function from NumPy's linear algebra module (np.linalg) to compute the inverse of a matrix, and the dot() method for matrix multiplication:

```
X_b = np.c_[np.ones((3320, 1)), X]  # add x0 = 1 to each instance
theta_best = np.linalg.inv(X_b.T.dot(X_b)).dot(X_b.T).dot(y)
Theta_best
```

```
array([[-0.0108267 ],
       [ 0.28716253]])
```

Predictions using ^θ

Now we can make predictions using ^θ:

```
X_new = np.array([[0], [2]])
X_new_b = np.c_[np.ones((2, 1)), X_new]  # add x0 = 1 to each
instance
y_predict = X_new_b.dot(theta_best)
Y_predict
```

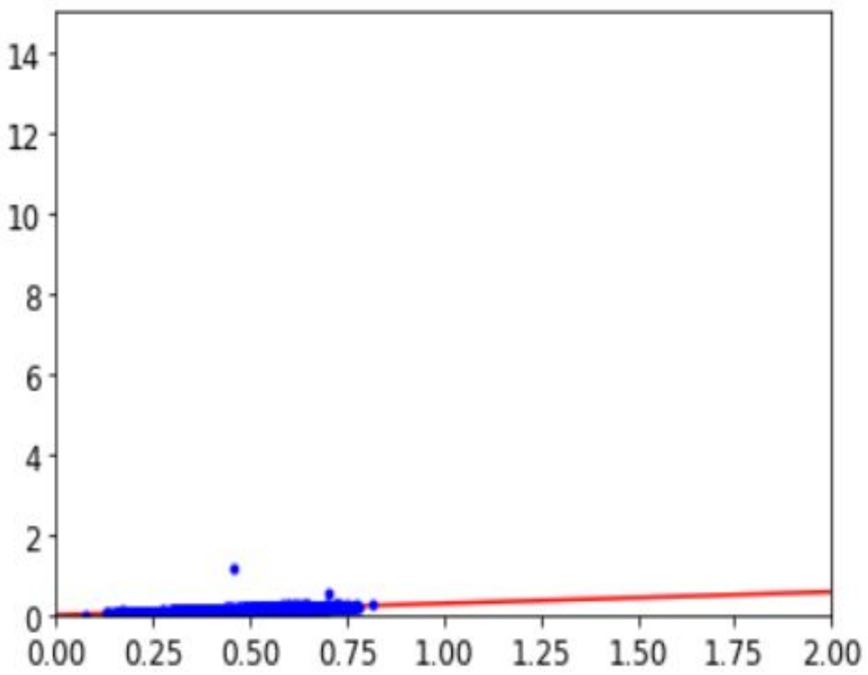
Output -

```
array([[-0.0108267 ],  [ 0.56349837]])
```

Plot model's predictions

Let's plot this model's predictions -

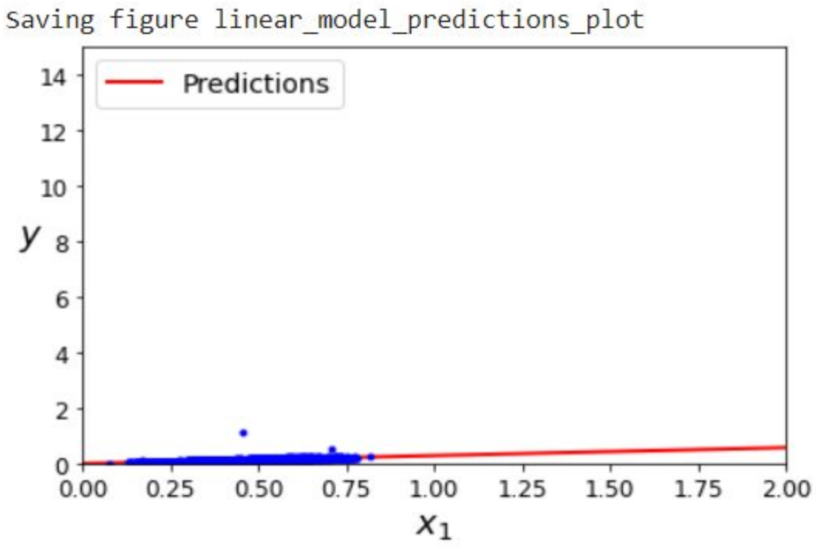
```
plt.plot(X_new, y_predict, "r-")

plt.plot(X, y, "b.")

plt.axis([0, 2, 0, 15])

plt.show()
```



Plot with Data legend and axis labels

The figure actually corresponds to the following code, with a legend and axis labels:

```
plt.plot(X_new, y_predict, "r-", linewidth=2, label="Predictions")
plt.plot(X, y, "b.")
plt.xlabel("$x_1$", fontsize=18)
plt.ylabel("$y$", rotation=0, fontsize=1
plt.legend(loc="upper left", fontsize=14
plt.axis([0, 2, 0, 15])
save_fig("linear_model_predictions_pl
plt.show()
```



Linear Regression using Scikit-Learn

Performing Linear Regression using Scikit-Learn is simple:

```
from sklearn.linear_model import LinearRegression
lin_reg = LinearRegression()
lin_reg.fit(X, y)
lin_reg.intercept_, lin_reg.coef_
(array([-0.0108267]), array([[0.28716253]]))
lin_reg.predict(X_new)
array([[-0.0108267 ], [ 0.56349837]])
```

Linear Regression using function and pseudoinverse

The LinearRegression class is based on the scipy.linalg.lstsq() function (the name stands for "least squares"), which you could call directly:

- theta_best_svd, residuals, rank, s = np.linalg.lstsq(X_b, y, rcond=1e-6)
- Theta_best_svd

array([[-0.0108267],[0.28716253]])

This function computes X+y , where X+ is the pseudoinverse of X (specifically the Moore-Penrose inverse). You can use np.linalg.pinv() to compute the pseudoinverse directly:

- np.linalg.pinv(X_b).dot(y)

array([[-0.0108267],[0.28716253]])

References

- https://npu85.npu.edu/~henry/npu/classes/machine_learning/book/hands_on_ml_with_schikit_2nd_edition/4.%20Training%20Models.html

- Google Slides URL -

  https://docs.google.com/presentation/d/1KXWju6cmLoX_l-dC2bUWbV5m086-uh8nUKRuu6n7vZY/edit?usp=sharing

- Github URL -

  https://github.com/santhinagalla/Machine-Learning/tree/main/Supervised%20Learning/Linear%20Regression